Implementation of Greedy and Backtracking

by Tedy Setiadi

Submission date: 02-Nov-2020 05:35AM (UTC+0700)
Submission ID: 1432983282
File name: Implementation_of_Greedy_and_Backtracking.docx (720.85K)
Word count: 1862
Character count: 9537

Implementation of Greedy and Backtracking Algorithm to Solve Rubik's Cube

Drs. Tedy Setiadi[1] & M. Noor Triasmara[2]
Department of Informatics Engineering
Universitas Ahmad Dahlan, Yogyakarta
E-mail: tedy.setiadi@tif.uad.ac.id

ABSTRACT

Rubik's cube is game mechanics which is useful for sharpening the brain. Nowadays. Many people who think that Rubik's Cube can be solved only by the people who have a high IQ. but already many algorithms found to solve the Rubik's Cube. so that anyone can solve it. In this paper, we explain how to solve the Rubik by applying the greedy algorithm combined with a backtracking algorithm. We solve this game by combining the greedy algorithm step by step followed by a backtracking algorithm. The advantage is this method is taster than the brute force method. In addition. this algorithm can be applied with the manual. and we can proceed with the development of the program. On its implementation to create 3-dimensional graphics facilities used open GL. The final results obtained with a solution rubik accompany every step instructions.

Keywords: Rubik, Greedy, Backtracking, program.

INTRODUCTION

Rubik's cube become phenomenally popular game that is often played by children and adolescents. because it requires speed and accuracy to match the colors of each side of the Rubik (Getsel,2005). Rubik's Cube is a mechanical game cube measuring 3 x 3 x 3 cube small. Each small cube side facing out has a different color. Rubik's Cube consists of twenty-six small cubes held together by a small cube centered Rubik's Cube. Each side of the Rubik's Cube can be played as much as 90°. 180° and 270° clockwise or counter-clockwise. At the beginning of the game Rubik's Cube rotated randomly so that each side will have side boxes with small cubes of different colors (See figure 1.a). Then it starts to be solved by the players. The game is completed Mien each side of the Rubik's Cube has checkered side small cubes of the same color (See Figure 1).

The combination of colors on a Rubik's Cube is very large. The first (possibly in the corner), the combination of each small box in the corner, there is a possibility 018 different places and at each place was a small box has 3 different colors, The second (possibly in the ribs), the small boxes which were in the ribs, there are 12 different places and at every place there are 2 different colors. While (possibly in the middle) the small box in the middle of each side will not change its place, there is only one possibility. So, there are many color combinations as following:

The number of combination = possibly in the comer X possibly in the ribs X possibly in the middle
= (8! X 38) x (12? X 212] x 1

= 519,024,039,293,878,272,000
= 5.19 x 10^{20}

Since the number of color combinations is available, but there is only one connect combination. then a lot of people who call the Rubik's Cube as the Magic Cube. They think that the cube can only be solved by magic (Gilles. 2006).

One possible “brute force” method to solve it would be determine once a sequence of moves which eventually reaches every possible position of the cube and then whenever we want to solve a cube. we just mindlessly follow the sequence until we eventually reach the solved cube. This strategy is absolutely failsafe, provided we have an extraordinary memory and enough time. Nevertheless there is still a lot of algorithms are used to solve it. And this paper we will discuss how to solve the Rubik's Cube using the Greedy algorithm in the first step and a step beyond using a backtracking algorithm. The reason is because this algorithm is easy to team and provides a solution relatively quickly compared with brute force algorithm (McMahon.2013).

Greedy Algorithm

Greedy algorithm is an algorithm that generates optimum solution through the completion of step by step (step by step). In the Greedy algorithm chosen option is the best option that can be obtained at the time regardless of the future consequences later (the take what you can get now). By choosing the best option at that time (local optimum) can achieve the best solution of problems faced (global optimum). In the Greedy algorithm assumed that a local optimum is a part of the global optimum (Weigang, 2010).

Optimization problem of the greedy algorithm composed by the following elements:

1. Candidate set. C set contains elements forming solution. At each step, the candidate is taken from its set.
2. The set of solution, S. Contains selected candidates as solutions to problems
3. Selection function - represented by the predicate SELECTION - that function that step of selecting the most probable candidate achieving an optimal solution.
4. Feasibility function - otherwise the predicate FEASIBLE - which is a function that checks whether a candidate who has been able to provide a feasible solution, le the candidate together with the set of solutions that have been formed not violate constraints. Feasible candidates put into set oi solutions. While unacceptable candidates discarded and never considered again.
5. Objective function. le functions that maximize or minimize the solution.

The greedy algorithm cm be implemented by the next function.

```
function greedy(input C: candidat set)→ candidat set
{return solution from optimization problem with a greedy algorithm
 input: the set of candidates C
 output: the set of solutions of the type set of candidate}
Deklaration
   x: candidate
   S: the set of candidate
algorithm:
    S ← {}  { initialize S with empty }
    while (not SOLUTION(S)) and (C ≠ {} ) do
    x ← SELECTION(C)          { select a candidate from C }
    C ← C - {x}          { element of the candidate set is reduced }
if FEASIBLE(S ∪ {x}) then
          S ← S ∪ {x}
    endif
```

```
endwhile
 {SOLUTION(S) or C = {} }

 if SOLUTION(S) then
   return S
 else
   write(' no solution')
      endif
```

BACKTRACKING ALGORITHM

Backtracking algorithms are search algorithms based on the DFS (Depth First Search). DFS algorithm is an algorithm used to find solutions for practical problems due to more quickly reach the depth of the search space (Wiliam. 2008). DFS only tracks algorithm that leads to a solution that will be processed and tracking solution will be based on space available solutions but not all of the solution space will be checked.

Backtracking algorithms in solving the overall problem requires a solution to the first problem, then other problems will be attempted to be resolved recursively based on the first solution. All solutions were made in the form of a solution tree (tree-shaped abstract) and the algorithm will traverse the tree develops problem solving DFS to find a viable solution. illustration of tree solution Backtracking algorithms described in Figure 1.

The process of the DFS search backtracking algorithm in Figure 1. do a search on all modes before searching a child node to the same layer. If the highest layer solution is not found then the search is continued on the node at the previous layer. The backtracking algorithm can be implemented by the next procedure.

```
procedure BacktrackingR (input k: integer)
{Finding all solutions to the problem of trace-back method; recursive scheme
 Input: k, the index of the vector component of the solution, x [k]
 Output: a solution x = (x [1], x [2], ..., x [n]) }
algorithm:
   for each x [k]such that untried (X [k] € T (k)) and B (x [1], x [2], ..., x [k]) = true do
        if (x [1], x [2], ..., x [k]) is the path from the root to the leaf then
                Print Solutions (x)
        endif
        BacktrackingR (k +1) {specify a value for x [k +1]}
   endfor
```


Object-Oriented Software Development

Object-oriented software development is a method of software development based abstraction objects that exist in the real world. Some object-oriented concepts used are: classes, objects, attributes. methods. messages. events and state. Encapsulation, inheritance and polymorphism are the main characteristics of the object that is programming oriented. Each concept can be used separately. but they complement each other synergistically. The concept of encapsulation allows software to develop faster and cheaper as well as meet the needs of the user (Larman. 2007). The object that wraps the data to communicate with each other, thus allowing reuse of certain codes (reusable components) as described in Figure 2.

RESULT AND DISCUSSION

Identification Of a Class (Object)

The classes that will build the software are as follows:

- C3DRubik class, a class that will define the format and properties of 3-dimensional Rubik use the OpenGL API functions.
- CWndRubik class, a subclass (inheritance) of class ClayerSolver and CbacktrSolver. This class is responsible for recording the input of Rubik.
- CrubikCube class. this class defines the Rubik properties such as color, size lighting and constituent cube rubik.
- ClayerSolver class, this class consists of methods and attributes to complete Rubik Greedy algorithm.
- CbacktrSolver class. the class that contains the method contains Rubik settlement with backtracking algorithms.
- CCtrtwarna class, contains methods for setting the display format 2D Rubik

The Design of Class Diagram

The Class diagrams to describe the relationships and interactions between class on making application rubik game is presented in Figure 3.

Results of the Program

Some screenshots of the program when the application run is shown as follows: Rubik random menu functions to work the entire layer Rubik randomly. so the colors of poles similar to the Rubik's are no longer seen in Figure 4.

The first step in solving the Rubik's backtracking algorithm method is to find the many small cubes that have color and the location which are appropriate. In completing each layer Rubik a small cube whose position has not been identified according to then be placed in a position that is supposed to be a small cube. In the completion of the first layer rubik, in addition to preparing small cubes, small cubes arranged are also located at the center of the second layer rubik that will become a reference in completing the second layer as shown in Figure 5.

The preparation ot the second layer of the cube refers to a small cube in the midst of completing a third cube layer. For side, bolted first reference. Pole of reference will be the root of the tree backtracking algorithm to solve the third layer. Reference pole in question is a green arrow as shown in Figure 6.

Greedy solving the menu is a menu for a settlement to resolve the Greedy algorithm rubik per layer. The first step of this method is to choose the first time that will be resolved and make the sides have similar color crossed where the colors for each end must be the same color as shown in Figure 7.

In completing the second layer, which is in the process ol preparing only a small cube that is at the end ol the second layer like Figure 8.

At the completion of the third layer. the method used to resolve is not like the first layer which form the first cross on the side of the layer as shown in Figure 9.

Figure Captions

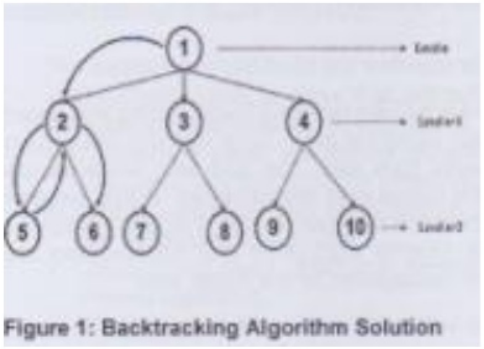

Figure 1: Backtracking Algorithm Solution

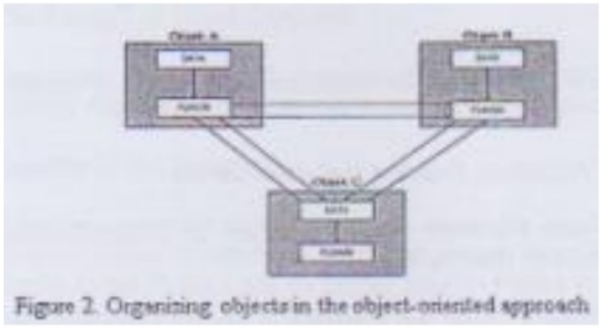

Figure 2. Organizing objects in the object-oriented approach

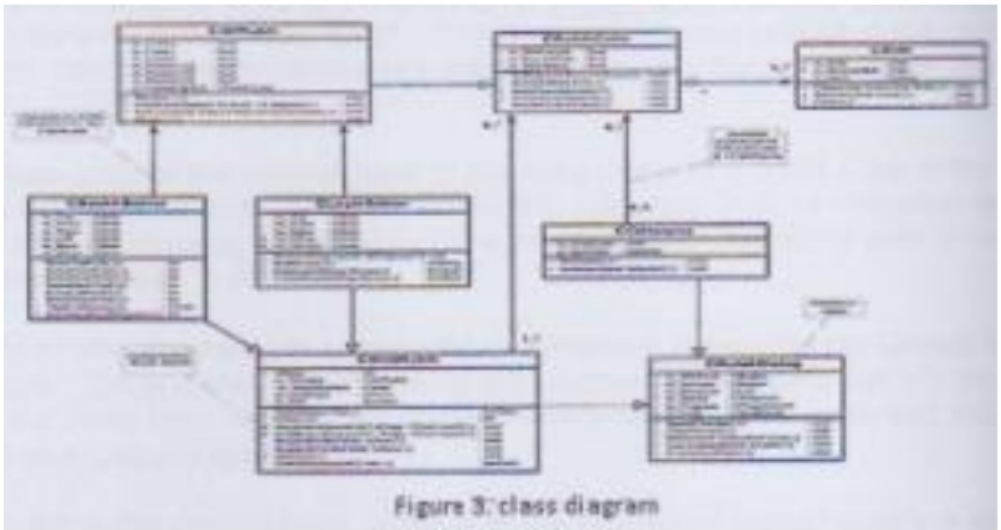

Figure 3: class diagram

Figure 4. Scrambled Rubik

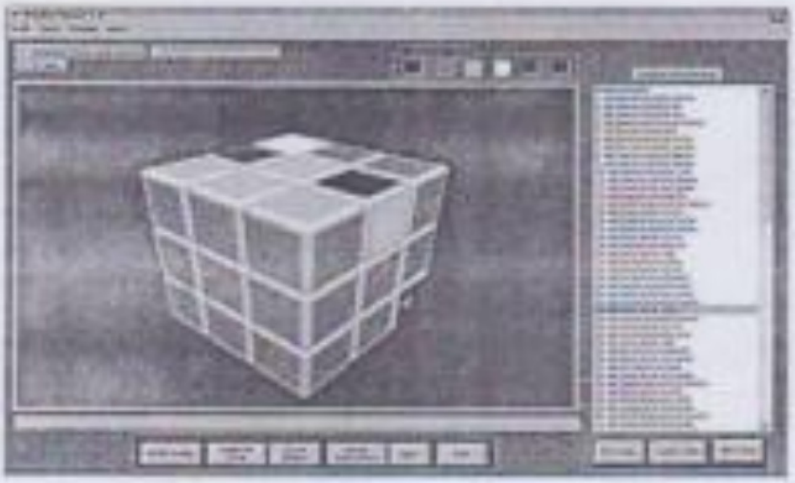

Figure 5. The first layer is completed backtracking method

Figure 6. Pole of reference for the completion of the layer 3

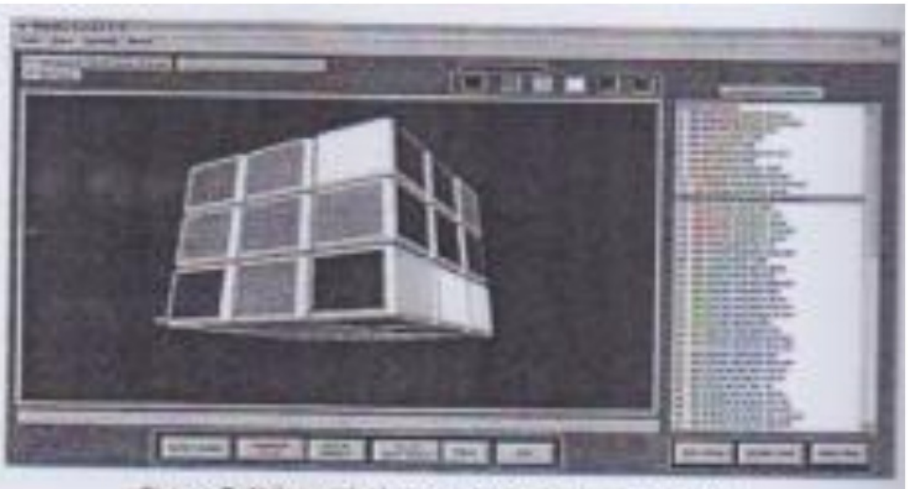

Figure 7. Sides with the same color intersect (cross)

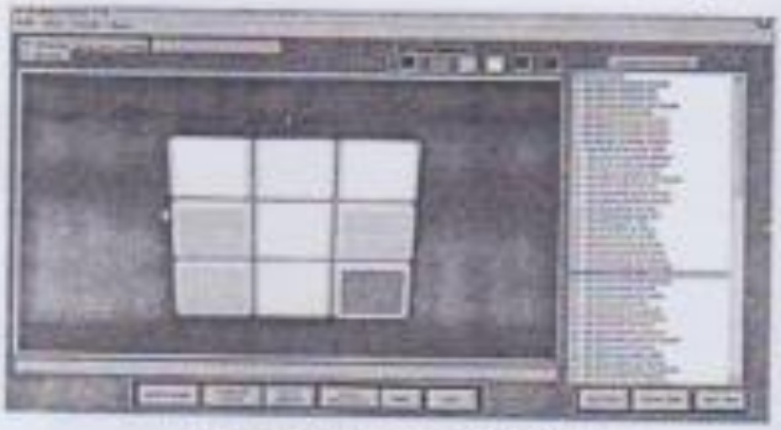

Figure 8. Small cube at the end of the layer 2

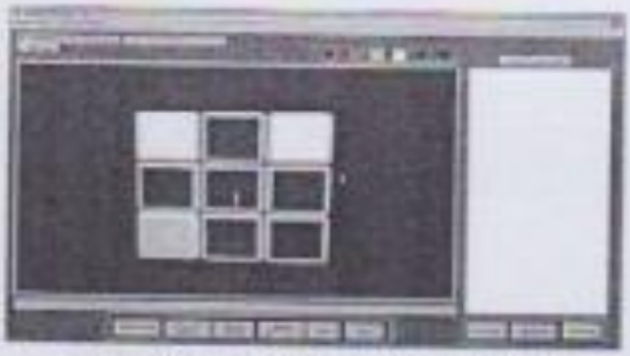

Figure 9. Cross on the side of the third layer

Completion Rubik Greedy algorithms in the program. will try all options of each side of the Rubik's solution to obtain the optimum step. Solutions on each side stored and compared, then which side chosen have the most optimum completion steps (minimum).

CONCLUSION

Although it looks difficult, but we can solve a Rubik's Cube by using backtracking and greedy strategies. These strategies are much better than using the brute force method. We can use a greedy strategy to solve Rubik layer by layer from step one and two, while we use backtracking in the third step to a node when it is raised ever raised before. In this research. we has successfully built a computer game application that can display 3 Rubik. Features or the software are menu Rubik settlement with Greedy algorithms and backtracking step instructions along with the solution, but it can be manipulated by the Rubik user.

REFERENCES

Auston Macmaho on http://snapguide.com/guides/solve-a+3x3-rubikscube-using-algorithms (access date 10-03-2014).

Getsel, Martin Van, 2005, "Cube", versi 1.42, Mach10 Software Production,. (*Freeware program*).

Gilles Brassard, 2006, Fundamentals of Algorithm, Prentice Hall, New Jersey.

Larman. Craig, 2007, "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design", Prentice-Hall Ptr, New Jersey.

Weigang. Jim, 2010, "Implementing Rubik's Cube: An Exercise in Hybrid Programming", Computer Language magazine.

William Khandar, 2008, "Layer by Layer Implementation of Algorithms tor Solving Rubik's Cube in the program code", paper STTEI ITB.

Implementation of Greedy and Backtracking

ORIGINALITY REPORT

7%	2%	6%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Huzain Azis, Rizaldi dg. Mallongi, Dirgahayu Lantara, Yulita Salim. "Comparison of Floyd-Warshall Algorithm and Greedy Algorithm in Determining the Shortest Route", 2018 2nd East Indonesia Conference on Computer and Information Technology (EIConCIT), 2018 Publication	5%
2	hdl.handle.net Internet Source	1%
3	Kartika Firdausy, KZ Widhia Oktoeberza. "Segmentation of optic disc using dispersive phase stretch transform", 2016 6th International Annual Engineering Seminar (InAES), 2016 Publication	1%
4	inba.info Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		